Lecturer Improvement Program at the National University of Singapore: A Software Engineering Approach

Elizabeth Vidal Duarte
Universidad Nacional de San Agustín, Perú evidald@unsa.edu.pe

***Abstract*— UNU-IIST contributes through Curriculum development projects, in which courses of software technology for universities in developing countries are developed and University development projects, which complement the curriculum development projects by aiming to strengthen all aspects of teaching in universities in developing countries. This paper shares the experience of the Lecturer Improvement Program in the National University of Singapore sponsored by UNU-IIST. We stand out the lessons learned about how to teach Software Engineering courses.**

***Keywords*—cross-cultural project; curriculum improvement**

Digital Object Identifier (DOI):
http://dx.doi.org/10.18687/LACCEI2017.1.1.195
ISBN: 978-0-9993443-0-9
ISSN: 2414-6390

Lecturer Improvement Program at the National University of Singapore: A Software Engineering Approach

Elizabeth Vidal Duarte
Universidad Nacional de San Agustín, Perú evidald@unsa.edu.pe

***Abstract*— UNU-IIST contributes through Curriculum development projects, in which courses of software technology for universities in developing countries are developed and University development projects, which complement the curriculum development projects by aiming to strengthen all aspects of teaching in universities in developing countries. This paper shares the experience of the Lecturer Improvement Program in the National University of Singapore sponsored by UNU-IIST. We stand out the lessons learned about how to teach Software Engineering courses.**

Keywords—cross-cultural project; curriculum improvement

I. INTRODUCTION

The United Nations University International Institute for Software Technology (UNU-IIST), established in Macau in 1992, was one of the Research and Training Centers of UNU (United Nations University) [1]. UNU-IIST had provided vital research on critical software systems while mentoring academics from around the developing world. Through its partner universities in developed countries, UNU-IIST provided off-shore fellowships which aims to strengthen teaching in universities in developing countries by training lecturers from these universities at partner universities in developed countries under the Lecturer Improvement Program [2].

This paper shares the experience of the author in the Lecturer Improvement Program (LIP) at the National University of Singapore (NUS). The objective of the LIP was to identify possible area that could be improved in the courses, curriculum and lectures in the Escuela Profesional de Ingeniería de Sistemas (EPIS) [9], Universidad Nacional de San Agustín(UNSA) [10] where the author is a full-time professor. The analysis was based on lessons taken from the attended courses at the National University of Singapore (NUS) and discussion with NUS staff.

After a review of the courses and make a comparison between the current courses taught in EPIS a series of lessons has been obtained. We point out a set of recommendations that were classified in three groups: recommendations that will help EPIS to improve courses, recommendations that will be helpful in improving lectures and recommendations for sharing this experience with most of the universities in our country. These recommendations were implemented in a medium and long term strategy.

Digital Object Identifier (DOI): http://dx.doi.org/10.18687/LACCEI2017.1.1.195
ISBN: 978-0-9993443-0-9
ISSN: 2414-6390

The rest of the paper is organized as follow section II explains the characteristics of the Lecturer Improvement Program and a brief description of the National University of Singapore. Section III describes the lessons learned and recommendations about course improvement. Section IV describes the lessons learned and recommendations about teaching improvement. Finally we share our conclusions.

II. LECTURER IMPROVEMENT PROGRAM

A. *Description*

Under the Lecturer Improvement Program (LIP) the lecturers from the developing countries generally spend one semester at one of the partner universities of UNU-IIST. During that time lecturers study several (generally four or five) courses offered by the partner university (Leicester, York, Belfast and Oxford in the UK, in Brisbane, Australia, Toronto in Canada and National University of Singapore in Singapore).

These courses may be at either undergraduate or postgraduate level. For each of these courses, they receive from the partner university all the appropriate course material (lecture material, student's notes, course exercises, etc.), and UNU-IIST provides them with recommended text book(s) for the course.

When the lecturers return to their home university, they use the knowledge they have gained on the project, together with the course material and text books provided to them, as the basis for improving existing courses or introducing new courses into the teaching curriculum of their own university, thereby updating and expanding this curriculum.

In order to maximize the benefits from the project for any particular developing country, the project is run on a "knowledge sharing" basis. It means that as far as possible the lecturers selected for the project from any given country study different sets of courses at overseas universities, and when they return home the knowledge and the course material and text books they acquire through the project are made available to other universities in the same country.

B. *National University of Singapore*

During the LIP the author was assigned to the National University of Singapore [3]. Specifically to the 4-years Bachelor in Computer Science [4] which is structured around the Association of Computing Machinery (ACM) and the IEEE

Computer Society's Computing Curriculum recommendations [5]. The curriculum structure is shown in Table 1.

TABLE I. NUS COMPUTER SCIENCE COURSES

Common Essentials (Mandatory)	
CS1101	Programming Methodology
CS1102	Data Structures and Algorithms
CS1104	Computer Organization
CS2102	Database Systems
CS2105	Introduction to Computer Networks
Major Requirements Computing Related (Mandatory)	
CS1231	Discrete Structures
CS2103	Software Engineering
CS2106	Operating Systems
CS3212	Programming Languages
CS3215	Software Engineering Project
CS3230	Design and Analysis of Algorithms
Elective I	
CS3211	Parallel and Concurrent Programming
CS3220	Computer Architectures
CS3231	Theory of Computation
CS3234	Logic and Formal Systems
CS3243	Foundations of Artificial Intelligence
Elective II	
CS4101	Honours Project
CS4214	Formal Semantics
CS4220	Comp. Analysis of Biological Data
CS4221	Database Design
CS4231	Parallel and Distributed Algorithms
CS4236	Principles of Computer Security
CS4237	Systems Modeling and Simulation
CS4241	Multimedia Information Systems
CS4243	Computer Vision and Pattern Recognition
CS4244	Knowledge-Based Systems
CS4247	Image Synthesis and Computer Animation
CS4248	Natural Language Processing
CS4250	IS Research Methodologies
CS4251	Strategic IS Planning
CS4252	Control, Audit and Security of IS
CS4253	Information Systems Consulting
CS4255	IT Outsourcing and Off shoring Management
CS4260	E-Commerce Business Models
CS4264	E-Commerce: B2C Applications
CS4266	IT and Customer Relationship Management
CS4271	Critical Systems and Their Verification
CS4272	Hardware-Software Co-Design
CS4274	Mobile and Multimedia Networking
CS4275	Programming Real-time systems
CS4343	Game Development Project
CS4344	Networked and Mobile Gaming

The first year of the degree courses introduces the fundamental concepts of programming, starting with well-structured object-oriented programs. They learn how to develop algorithms and efficient data structures to solve problems. In the second year students are provided with the basic understanding and appreciation of the various essential programming languages constructs. Students learn that software engineering is also about programming in the large by carrying out a "simple" project that covers all the phases of software development. In the third and fourth years students are able to select different courses in order to specialize themselves. Students have to take a minimum of two courses from the Elective I section in Table 1. Additionally, either they have to take CS4101 Honours Project and four courses at Elective II section, or to take seven courses at Elective II section.

According to the specifics needs of EPIS determined by discussion with the Head of EPIS, EPIS's Faculty and recommendations provided by the supervisor assigned by NUS, the attended courses were: Programming Methodology, Software Engineering, Software Engineering Project, Programming Languages Concepts and Programming Languages.

This paper present a detailed description of the courses related to the Software Engineering area. Additionally, we analyze the key features of these courses. As a result of the analysis we present the valuable lessons that were taken as guideline for EPIS curriculum and lecturer improvement.

III. NUS SOFTWARE ENGINEERING COURSE DESCRIPTION

A. Software Engineering (CS2103)

This course introduces the necessary conceptual and analytical tools for systematic and rigorous development of software systems. It covers four main areas of software development, namely object-oriented analysis, object-oriented design, implementation, and testing, with emphasis on the design and implementation of software modules that work cooperatively to fulfill the requirements of a system.

Tools and techniques for software development, such as Unified Modeling Language (UML), program specification, and testing methods, are taught too. Major software engineering issues such as modularization criteria, program correctness, and software quality are also covered.

The course is considered as an introductory course. Topics covered include an overview of software process, methodology and software development models. Requirement phase: use case and domain modeling and activity diagram. Analysis phase: software system architecture, use case realization, class and object models. Design phase: class and object diagrams, interactions diagrams, state-charts, design patterns. Implementation and Testing phase: unit testing, integration testing and automated test driver.

The final aim of this course is to produce a "simple" software. It is done in tree submissions (1) Requirement and Analysis, (2) Design and (3) Implementation and Testing.

The structure of the course is shown in Table II.

TABLE II. CS2103 SOFTWARE ENGINEERING

Software Engineering		
Year	2	
Credits	4	
Prerequisites	CS1102	
Assessment	Midterm exam: 20% Tutorials: 5%	Project: 25% Final exam: 50%
Lectures	13	
Lectures Hours	2 hours per week	
Tutorials	2 hours per week	
Lecturers	Dr. Soo Yuen Jien and Dr. Damith C. Rajapakse	
Webpage	Not Accessible	

The objective is to teach the understanding and use of object-oriented methods to analyze, specify design and implement large computer systems. Lectures gives the theory background. Tutorial allows students to explore the theory. Project is a tool for students to apply what they have learned.

B. Software Engineering Project(CS3215)

In this course, students design and implement a tool called Static Program Analyzer (SPA). SPA is an interactive tool that automatically answers queries about programs. It tries to locate code relevant to the maintenance phase.

Project stages include analysis and architectural design and three iterations in which students develop SPA incrementally. The details of what students should deliver in each stage are described in five assignments. The SPA project is developed in teams of 4-6 students. Each team is further divided into two groups of 2-3 students. Each group delivers a different subsystem of the SPA. These subsystems communicate through a non-trivial interface. While each group will have to complete an independent piece of work, groups in a team will have to communicate a lot to integrate their work and to get the SPA product right. The structure of the course is shown in Table III

TABLE III. CS3215 SOFTWARE ENGINEERING PROJECT

Software Engineering Project	
Year	3
Credits	8
Prerequisites	CS2103
Assessment	20% - Assignments 1, 2, 3 and 4. 80% - Final Project report and demo of the program.
Lectures	5
Lecture Hours	2 hours per week
Tutorials	2 hours per week
Lecturers	Dr. Stan Jarzabek and Dr. Damith C. Rajapakse
Webpage	http://www.comp.nus.edu.sg/~cs3215/

The objective is that students work through the Software Development Life Cycle (SDLC) to complete a team project. Specific objectives are: To prepare students for industrial projects, to develop the ability to work in groups in a project of substantial size and complexity, to enhance project planning skills, to develop communication and writing skills, to apply and consolidate what students have learned in CS1101, CS1102 and CS2103 and To follow the SDLC according to the "best software engineering practices", to develop a well-tested production quality software system.

Regarding to the method class there are only 5 lectures in this course. During the lectures, students discuss the team project “Static Program Analyzer” from the functional, architecture and technical point of view and explain the SDLC for the project. There are five assignments leading to project completion. In the last assignment, at the end of the course, students submit a final project report and demo the system they have implemented. For each assignment, students submit a documented solution and present their solution during tutorial hours.

IV. COURSE IMPROVEMENT RECOMENDATIONS

After attending the courses for a whole semester and continues discussions with NUS staff important lessons and recommendations have been obtained.

A. Improving Software Engineering Courses

EPIS should cover the topics of the software development process in a single course namely Software Engineering I. Topics must include: software engineering principles, software development process, object-oriented requirement capture and analysis, object-oriented design, implementing and testing. In order to give students a practical experience about how models become into code it is necessary to implement a small project.

EPIS should also include a course like the Software Engineering Project (CS3215) from NUS. In this course, students design and implement a tool called Static Program Analyzer (SPA). SPA is an interactive tool that automatically answers queries about programs. It tries to locate code relevant to the maintenance phase.

Project stages include analysis and architectural design and three iterations in which students develop SPA incrementally. The details of what students should deliver in each stage are described in five assignments. The SPA project is developed in teams of 4-6 students. Each team is further divided into two groups of 2-3 students. Each group delivers a different subsystem of the SPA. These subsystems communicate through a non-trivial interface. While each group will have to complete an independent piece of work, groups in a team will have to communicate a lot to integrate their work and to get the SPA product right.

This course follows the Software Engineering course (CS2103) where students got a broad view of the software development process. In CS3215 students develop a software tool, with much emphasis on architecture, complex design problems, data structures, algorithms and incremental development.

Since it is an eight-credit course, students are expected to spend the equivalent time to two courses on the project work. Time management is a critical success factor for this course. Attitude “just get a program run” doesn’t work on this course. They have to consider reliability, high quality of documentation and report, flexibility and reusability. Also they have to demonstrate some degree of innovation in terms of features and design solutions and demonstrate maturity of skills in areas of team work, design, incremental development, and testing.

As mentioned in the course description, each team project is further divided into two groups that deliver a different subsystem of the SPA. As the team progresses into the project, students create more and more source code. As each of them is working independently on different parts of the system, they occasionally need to pass their set of codes to other members for integration. Students are strongly encouraged to use control-version system software. Students are required to test their software during the development process. How much testing they need to do depends on how much they need to verify the code. In order to facilitate the project development NUS uses CVS [7] and CPP Unit [8] tools.

It is interesting for our analysis to observe how NUS students put into practical use the techniques of Software Engineering that have been studied so far. But additionally, this module is indeed a good practice for students to learn about software project management. Student experience that time management is extremely important during the development of the project. It is the responsibility of every member to adhere strictly to any deadlines that have been set. Since modular programming is adopted as a practice, all the modules are not exactly independent of one another. Hence, it would be impossible to integrate the entire project together if any one of the modules were not ready. It is very important that every member in the group cooperates and communicates extensively. Another point is that students have to plan their working schedule for this project. Tasks should be distributed clearly, and an agenda should be planned for the group meetings to achieve effective, focused discussion.

V. Teaching Improvement Recomendations

A. Covering more body of knowledge

One of the features of world-class universities is that they use less lecture-hours per-week to cover the body of knowledge. The average time assigned is two lecture-hours and two tutorial-hours per course – per week. For the purpose of this recommendation, we will focus on why our lecturers do no cover all the topics considered in the courses they currently teach, considering that our curriculum has twice as much the assigned hours per course-per week.

One of the best practices of lecturing at NUS was that Lecture notes and lecture slides ought to be uploaded to the course web-page at least two days before the lecture. Students must read the lecturer notes and bring them printed to class.

This practice allows lecturers to cover more body of knowledge since students do not spend much time taking notes. Additionally, students get a general idea about the topics before the lecture.

B. Filling the Gap between Theory and Practice

Tutorials and Labs form the practical element of NUS courses. NUS attaches great importance to developing good programming skills in students. NUS closely monitors students’ progress through the tutorial hours. The tutorials are used to discuss the assignments given in class. It is not enough to work on the assignments during the lab or tutorial sessions. NUS students have to work on the assignments before the lab sessions. They use the lab to have a focused question-answer session about the assignments.

In order to reinforce and ensure students learning, we suggest some changes and a new organization in the laboratory sessions at our university:

- The assignments should be proposed by the lecturer in charge of the theory
- To have a better communication with the lab staff in order to make sure that assignments are strongly related to the theory.
- To encourage students to make more challenging assignments.
- To make sure students use the lab hours to have focused answer-question sessions.

C. Using Technology

NUS lecturers make intensive use of technology in their courses. Every course has an official webpage. Additionally, they make use of and Integrated Virtual Learning Environment (IVLE). This web environment allows NUS lecturers to post their notes, announcement, assignments, forum discussions, etc. We can assure that 100% of NUS lecturers make good use of IVLE or course web pages.

Since 2015 UNSA has been working with Google ClassRoom. The use has been very light; just some lecturers have been using it to upload their lectures notes. But these notes have been posted after the lecture sessions in most of the cases.

The use of technology will allow our lecturers to cover more body of knowledge. Since there is already a Virtual Learning Environment tool lecturers should upload their lecture material before the class session.

NUS students are encouraged to use IVLE Forum. Forums are good for understanding the course material since students can post example codes or test programs. Forums open discussions about course materials and assignments that were not clear enough. It is a vehicle for sharing information and helping each other. It allows lecturers to offload a lot of subsidiary discussions out of the classroom.

Forums are also good platforms to gauge how much students have understood the course materials, and what problems they are facing. Also lecturers will be able to check whether students have done the reading assignments or exploratory questions that lecturers have told them to do.

VI. SHARING RECOMENDATIONS

Many universities in Peru suffer from isolation from the international academic community: mostly because they have very little money available for international travel. This makes it very difficult for universities to keep abreast of advances in the subjects they teach, particularly in a field such as software engineering which changes so rapidly.

In order to maximize the benefits from this experience, it was necessary to share the lessons obtained from NUS with any university in Peru. It means to share the course materials as well as the knowledge obtained in the new courses studied. Additionally, it was important to point out the best teaching practices learned in a world-class university like NUS. For these purposes we proposed a series of seminars for lecturers in our University and other two Universities in our city. These seminars considered the following topics: (1) Teaching the Capstone Project in Software Engineering (2) General Teaching and Research experiences in a World-Class University.

Conclusions

This paper has presented the experience in the Lecturer Improvement Program sponsored by the UNU-IIS. We have suggested a series of recommendations in order to improve Software Engineering courses and teaching skill. The recommendations were based on the courses attended for the author at the National University of Singapore and discussions with NUS faculty.
From the comparative analysis made between courses at NUS and courses at EPIS, our analysis has shown that there are some core-units missing in Software Engineering courses. Also our analysis pointed out that it is necessary to reassign some units in the Software Engineering course in order to give students a deeper theory and practical understanding of the software development process.

Additionally, we have presented a series of recommendations related to the improve teaching. We have proposed the intensive use of an Internet Learning Environment. This tool will help lecturers to cover more body of knowledge by uploading the material before the class.
All the recommendations were. After ten year of the fellowship we are still in touch with NUS faculty. They share new material with us and they keep us updated about new material for the courses.

Finally, for nearly a generation, UNU-IIST has been committed to providing vital research on critical software systems while mentoring exceptional academics from around the developing world. As the world of information and communication technology (ICT) has changed immensely over the decades, it has defined a new mission along with a four-year strategic plan in 2010. The plan calls for UNU-IIST to embark on a more dramatic and focused response to the new computing environment and its potential to serve the cause of sustainable development. UNU decided to evolve the former IIST into a new Institute on Computing and Society (ICS). The mission of UNU Computing and Society (UNU-CS) is to focus on the key challenges faced by developing societies through high-impact innovations in computing and information technologies.

ACKNOWLEDGMENT

The author is deeply grateful to UNU-IIST for giving her the opportunity to be part of the Lecture Improvement Program at the National University of Singapore
Sincere thanks to Dr. Soo Yuen Jien and Dr. Damith C. Rajapakse whom gave me the opportunity to learn a better way of teaching. The author is deeply grateful to her supervisor at NUS Dr. Chin Wei-Ngan, for his extensive support, discussions and guidance throughout her work.

REFERENCES

[1] United Nations University https://unu.edu/
[2] Z. Chaochen, How UNU/IIST Serves Developing Countries?http://unpan1.un.org/intradoc/groups/public/documents/apcity/unpan001464.pdf
[3] National University of Singapore. http://www.nus.edu.sg/
[4] Bachelor of Computer Science – National University of Singapore. http://www.comp.nus.edu.sg/programmes/ug/cs/
[5] ACM/IEEE Curriculum Guidelines for Undergraduate Degree Programs in Computer Science. https://www.acm.org/education/CS2013-final-report.pdf
[6] Java Programming Style, Design and Marking Guidelines http://www.comp.nus.edu.sg/~cs1101x/3_ca/labs/styleguide/styleguide.html
[7] CVS - Concurrent Versions System. http://www.nongnu.org/cvs/
[8] CppUnit. https://sourceforge.net/projects/cppunit/
[9] Escuela Profesional de Ingeniería de Sistemas http://www.episunsa.edu.pe
[10] Universidad Nacional de San Agustín http://www.unsa.edu.pe